# INTRODUCTION

## Purpose

The purpose of this book is to help new Harmony users climb the steep learning curve so they can use the MPLAB Harmony development environment to develop reliable and reproducible applications using Microchip devices. The "learn-by-doing" method used in the book provides a deep understanding of the underlying structures and architecture of Harmony which can be applied to other starter kits or custom boards.

The Harmony development environment is the preferred method for software development using the MX and MZ products from Microchip. There are older methods of programming these devices, but Harmony is needed if the developer wants to include the USB library or TCP/IP stack when developing new applications with the MZ product.

## Why This Book?

Microchip provides a wide variety of hardware products supported by a comprehensive suite of software development tools. For newbies, however, choosing which tools and products can be overwhelming. The Microchip starter kits can be used to demonstrate the hardware and software capabilities, but the learning curve can be daunting. Even though Microchip provides wikidot tutorials and thousands of pages of documentation, there is little or no guidance to explain the "how and why" when developing Harmony applications

I have decomposed some of the basic Harmony examples and reconstructed them as an integrated application using the PIC32MZ(EF) Starter Kit hardware. There are five complete projects in this book that demonstrate: the use of timers to control LEDs and switches; the use of the USB Communications Device Class (CDC) library to communicate with a PC; the use of the TCP/IP stack to develop an http_net web server for an embedded system; and how to convert two of these projects into FreeRTOS projects.

## What is Different from The MX Book?

My first book, "Harmony for PIC32MX Applications", is based on the Ethernet Starter Kit II (MX chip) and targets the new-to-Harmony user. This book is also for a new Harmony user, but it addresses functionality available with the more powerful MZ chip and includes features that are of interest to the intermediate-level developer.

I have added new sections to this book that: (a) use the Pin Manager to connect the board peripherals to the application code, and (b) added an entire chapter that addresses how Harmony can be used in conjunction with the real-time operating system, FreeRTOS.

## How Is This Book Different From Other Learning Approaches?

After decades of using a variety of software tools for mainframes, minis, micros and devices, I have always found the best starting point to be the "Hello world" step. [In the case of the MZ EF Starter Kit, the equivalent of the text message, "Hello world", is to blink the LEDs.] Unfortunately, there is no "Hello world" example using Harmony nor is there a simple way to get started.

The suggested approach to learning Harmony is to install one of the Microchip demo examples provided with the Harmony installation package. But which ones? Some use Harmony, some do not. Not all examples work on all starter kits/evaluation kits. Worse yet, after installing an example project, if any changes are made, you could generate a screen full of errors and not know why or how to fix them without a deeper understanding of the software structure. Thus, the demo-based learning is unacceptable (to me) because there is little or no explanation of the underlying structures and architecture.

Another learning approach is to go to the online forums and follow some of the threads to understand the processes and software. That approach will not work. Online forums are very specific about single issues. Expecting to piece together forum threads into a cohesive, meaningful project is impossible (IMHO).

A third learning approach is to read all the documentation and re-create one or more of the Microchip examples from scratch. Although there are volumes of documentation, not all of it is useful or of consistent quality. There are many examples of document frustration on the user forums. Viewing the Harmony documentation as a learning source is a very inefficient way to develop real applications (IMHO).

All of these learning approaches are attempts at a short cut to what is really needed – a bottoms-up method that builds on the knowledge and experience of previous steps.

## Projects Based Learning

Please note: This book assumes the reader has a working knowledge of real-time systems and is proficient in the C programming language.

A tutorial approach is used to build MPLAB "projects" that results in high quality Harmony applications that provide a deep understanding of the principles involved in embedded systems programming. Following the steps in each tutorial demonstrates how to use Microchip hardware and software in a reliable and reproducible way. Reproducibility is especially important using the Harmony environment because the components and tools are constantly evolving which can change the results of previously developed software using MPLAB X.

The methodology used in this book establishes a base project and then builds on that base to add complexity, leading to a powerful TCP/IP application. This "building block" approach allows the reader to include, exclude, or modify the components to fit a specific need.

## Book Structure

This book is organized as follows:

Introduction -- Explains the purpose, the reason for the book, and the benefits from using a projects-based approach to learning.

Chapter 1 -- Develops the Harmony environment and shows how to download, install and execute the TCP/IP demo example project. Once the hardware and software are functional, a new base project is created from scratch using a minimum of Harmony components. The base project demonstrates how to compile, load, and download the project to the target device (the MZ EF Starter Kit), followed by how to debug the project using the MPLAB X tools.

This chapter also explains the key components of Harmony, such as the superloop and file structures, that are used in every Harmony project.

Chapter 2 -- Extends the base project to provide a rudimentary user interface (UI) using the starter kit hardware components. The use of timers to control the LEDs and switches is demonstrated. At the end of this chapter, the base project is complete and can be used as a starting point for other applications. By the end of

Chapter 2, the reader should be comfortable with the Harmony environment and its capabilities.

Chapter 3 -- Demonstrates how to use the Harmony USB library for a USB CDC application. The USB communications device class (CDC) is added to the base project to allow the application to communicate with a PC over two serial ports (COM ports). One COM port is dedicated to collect and display text messages (log messages) from the application while the other is available as a console port to send commands to the application.

Chapter 4 -- Creates a new project cloned from the previous chapter's project. This new project is used to show how to configure and use the Harmony TCP/IP stack to develop an http and https web server that serves the Microchip example web pages.

Chapter 5 -- Shows how to create and download custom-designed web pages using the mpfs2.jar utility program (provided with Harmony). This chapter also shows how to process dynamic variables for the custom web pages rather than the variables used in the demo example pages.

Chapter 6 – Explains how the real-time operating system, FreeRTOS, can be integrated into a Harmony project. Two examples in this book are updated to use FreeRTOS rather than the superloop for real time control of the application.

Chapter 7 -- Provides some useful tools and techniques that an embedded designer may find useful. The two ancillary techniques introduced in this chapter include: (1) downloading compiled code without the need for the entire MPLAB X development environment, and (2) simple configuration management for MPLAB X projects.

## Benefits

### Reproducible

The projects in this book have been developed using step-by-step procedures that provide a way to go back to previous project milestones to see the results of a particular step before the project takes a wrong turn. Many of the steps show the screen shots of the MPLAB Harmony Configurator (MHC) settings in order to understand how MHC affects the application. MHC is a major component needed to develop Harmony applications.

### Building block approach

Although Harmony uses "modules", problems can arise because of interdependencies leading to errors and frustration. For example, adding a module (e.g. SYS_CONSOLE) in MHC can bring in other modules that may interfere with your application. Then taking out that module can leave traces in the application that have to be detected and eliminated by debugging. In contrast, all of the building blocks used in this book are well-defined and can be added and deleted without harmful side effects.

### Improved learning

Using an environment such as Harmony involves much more than the libraries and C-code. As a newbie, the entire MPLAB X IDE presents a steep learning curve just to get to the simple "Hello World" project. The step-by-step instructions/procedures included in this book provide immediate learning of the tools needed to develop a complete application. Explanations of the "why" and "how" for source code included in a step are included to increase the understanding of Harmony.

### Obsolescence

A major problem with writing a book about Harmony is the concern that the contents will be obsolete by the time the book is published. There is no question that the contents of Harmony and its toolset do undergo changes, but please understand that Harmony is an amorphous collection of components, many of which are stable. This book is based on the stable modules used in most embedded systems. Although the screen shots contained in the book may have slightly different looks as MPLAB X evolves, the functions and results will remain the same. The same comment applies to a PC or MAC host – the results are the same even if the screen shots have slightly different graphics symbols.

### Faster Debugging Time

In any software development project, finding and fixing "bugs" is a major source of frustration and project schedule delays. Although there are hundreds of pages of Harmony-related documentation, YouTube videos, and online forums, debugging is still an art that requires deep understanding of the system and its capabilities. For Harmony, one of the best debugging tools is the demo projects distributed with each update of the system. If there is a demo project that uses the same components as a custom project, then comparing source code, settings and results makes debugging that much easier.

For the USB library and http_net modules, having an additional working, documented example can help identify and fix bugs more quickly. Just transposing the examples in the MX book to the MZ chip took almost three weeks of debugging to find some very subtle and undocumented “features” in those libraries. I expect any new custom application will have similar problems but if they can be found and fixed quickly using information in this book, then I have succeeded.